From Academia to Real World : a Practical Guide to `Hitag-2` RKE System Analysis

Ryad Benadjila[1], Mathieu Renard[2], José Lopes-Esteves[2], Chaouki Kasmi[2]
[1] ryadbenadjila@gmail.com [2] prenom.nom@ssi.gouv.fr

ANSSI

Résumé. Depuis 2006, plusieurs articles ont été consacrés à l'analyse de la sécurité de l'algorithme `Hitag-2`, notamment utilisé dans le contexte de contrôle d'accès aux fonctionnalités de véhicules comme l'anti-démarrage et l'ouverture. Bien que cet algorithme soit reconnu comme faible depuis longtemps, il semblerait qu'il soit encore utilisé dans l'industrie automobile. Dans un article récent paru à Usenix 2016, de nouvelles vulnérabilités concernant les systèmes de déverrouillage distant (RKE, pour *Remote Keyless Entry*) ont été exposées. Grâce à une nouvelle cryptanalyse de `Hitag-2` adaptée au contexte RKE, les auteurs peuvent forger une fausse clé permettant de (dé)verrouiller une voiture cible en capturant préalablement 4 à 8 trames radio.
La présente étude s'inscrit dans le cadre de ces publications, afin de qualifier le niveau de menace des vulnérabilités publiques dans un contexte d'attaquant boîte noire (c'est-à-dire sans accès aux calculateurs des véhicules ou à des documents non publics). La découverte de divergences avec l'état de l'art a nécessité un travail d'adaptation non négligeable dont les détails sont présentés dans cet article. Une nouvelle attaque efficace contre les systèmes RKE basés sur l'algorithme `Hitag-2` est proposée.

1 Introduction

Aujourd'hui, les constructeurs automobiles tendent vers l'interconnexion des réseaux embarqués des véhicules à différents systèmes d'information : système de gestion des défauts, de maintenance, d'urgence, etc. L'ouverture de ces réseaux au monde extérieur amène dès lors des problématiques de sécurité habituellement étudiées dans des écosystèmes plus classiques. Jusqu'à présent, les calculateurs ECU *(Electronic Control Units)* embarqués dans les véhicules étaient conçus uniquement en prenant en compte la sécurité au sens sûreté de fonctionnement. Le système d'ouverture à distance du véhicule constitue un maillon potentiellement faible d'un ensemble de mécanismes de sécurité quasiment inexistants.

Bien que l'algorithme `Hitag-2` soit identifié comme vulnérable depuis 2006 [7, 8, 16, 29, 31], celui-ci est toujours utilisé et fait l'objet de nouvelles

publications, notamment du fait de son usage dans les systèmes d'anti-démarrage des voitures [28]. En 2016, Garcia *et al.* présentent une nouvelle cryptanalyse de cet algorithme [12]. Cette fois, l'attaque ne cible plus les fonctionnalités d'anti-démarrage des véhicules, mais le système d'ouverture à distance. L'attaque permettrait à un attaquant capable d'enregistrer 4 à 8 trames radio de retrouver la clé de chiffrement servant à protéger les échanges. Cet attaquant pourrait alors réaliser un clone du transpondeur pour déverrouiller le véhicule cible.

Afin de qualifier le niveau de risque réel, cette étude a été réalisée dans le rôle d'un attaquant en boîte noire n'ayant accès ni aux calculateurs ni aux documentations privées des composants (hormis celles librement accessibles, qu'elles soient publiques ou qu'elles aient été divulguées par ailleurs).

L'article se décompose comme suit : la section 2 présente les différentes étapes préliminaires à l'étude, dont les éléments nécessaires à la compréhension des systèmes (RKE pour *Remote Keyless Entry*) : modèle d'attaquant et concepts radio ainsi que l'état de l'art des attaques sur `Hitag-2`. La section 3 a pour objectif de donner au lecteur les connaissances de base nécessaires à la compréhension de l'analyse des signaux radiofréquences générés dans le cadre des RKE. Les divergences de résultats obtenus ainsi que les mécanismes de défense identifiés sur notre véhicule de test, qui rendent les attaques de l'état de l'art difficilement applicables dans notre cas, sont présentés en section 4. Les résultats d'une nouvelle cryptanalyse fondée sur une recherche exhaustive optimisée et adaptative, permettant à un tiers malveillant d'ouvrir et fermer un véhicule après avoir capturé seulement deux à trois trames radio, et ce malgré les mécanismes de défense identifiés, sont détaillés en section 5. Enfin, les possibles contremesures sont discutées en section 6.

2 État de l'art et contribution

2.1 Rôles et usages des clés automobiles

Les clés de voitures modernes sont généralement utilisées pour deux fonctionnalités complémentaires :

– l'**anti-démarrage** (*immobilizer* en anglais) utilise des technologies de radiocommunication en champ proche de type RFID (généralement à une fréquence porteuse autour de 125 kHz) pour détecter la présence d'une clé valide et débloquer l'anti-démarrage. La portée est de quelques centimètres, et le barillet du verrou de démarrage contient une boucle

de champ proche (antenne). À la mise du contact (lorsque la clé est insérée et tournée), un protocole d'authentification entre la clé et l'unité de commande électronique en charge de la gestion du contrôle d'accès physique, est déclenché. L'ECU ne débloque la pompe à injection de carburant que si une clé valide est authentifiée ;
– l'**ouverture/fermeture** à distance RKE permet une ouverture des portes du véhicule par un appui sur un bouton présent sur la clé. Cet appui déclenche une transmission unidirectionnelle sur un canal radiofréquence d'une des bandes ISM (Industrielle, Scientifique et Médicale) ultra hautes fréquences (UHF), autour de 433 MHz et 868 MHz, avec une portée de quelques dizaines de mètres.

Ce sont des composants électroniques nommés **transpondeurs** qui implémentent ces fonctionnalités dans les clés. La cryptographie utilisée pour ces deux fonctionnalités peut être différente ou non en fonction des constructeurs ou des équipementiers. Des attaques cryptographiques combinées peuvent être élaborées lorsque les deux systèmes utilisent la même clé, permettant alors d'ouvrir et de démarrer la voiture [12]. Sur les nouvelles générations de véhicule, les systèmes RKE sont progressivement remplacés par des systèmes dits PKE *(Passive Keyless Entry)* et PKES *(Passive Keyless Entry and Start)*. Sur ces systèmes, l'ouverture et/ou le démarrage sont contrôlés via un mécanisme de détection de proximité.

Dans la suite de cet article, seule **l'analyse d'un système RKE** classique utilisant le protocole `Hitag-2` dans un contexte d'attaques permettant d'ouvrir la voiture sans la clé d'origine est abordée.

2.2 Systèmes RKE

Un système RKE est un système d'accès électronique qui permet de contrôler un mécanisme d'ouverture/fermeture à distance. L'action d'un utilisateur amène une clé (physique ou logique) à transmettre un signal radio à un récepteur qui contrôle le verrou électronique du système. L'action consiste à appuyer sur un bouton d'une télécommande physique ou d'une application mobile.

Modèle d'attaquant Un système RKE est sujet aux attaques inhérentes aux transmissions sans fil :

– l'interception : la propagation du signal n'étant pas maîtrisée, un attaquant disposant de moyens de réception adéquats peut recevoir, enregistrer, décoder et analyser les trames émises ;

- le brouillage : un attaquant disposant de moyens d'émission de signaux radiofréquence peut générer un signal interférant avec le signal légitime et entravant à son interprétation par le récepteur légitime ;
- le rejeu : un attaquant disposant de moyens d'émission et de réception peut intercepter des trames légitimes et les transmettre *a posteriori* au récepteur légitime.
- l'usurpation de source : un attaquant disposant de moyens d'émission et de réception (ou une connaissance des trames légitimes) peut forger et émettre des trames valides à l'attention du récepteur ;
- le relais : un attaquant disposant de moyens d'émission et de réception peut, lorsque l'émetteur légitime est hors de portée du récepteur légitime, acquérir les trames émises par l'émetteur et les relayer en temps réel au récepteur sans les modifier.

Plusieurs travaux de recherche et preuves de concept ont illustré l'impact de ces menaces sur les systèmes RKE et analogues. Citons notamment :

- dans [10], les auteurs démontrent la faisabilité des attaques par relais sur des systèmes PKE. Bien que leur étude se soit limitée aux communications basses fréquences en champ proche des PKE, le même principe est applicable aux communications UHF des RKE ;
- une attaque sur des RKE par brouillage *(jamming)* est proposée dans [24]. Le principe de l'attaque repose sur la possible inattention de l'utilisateur lorsqu'il ferme son véhicule. Le principe est de brouiller la transmission radiofréquence lorsque l'utilisateur presse le bouton de verrouillage du véhicule. Le véhicule reste ainsi ouvert malgré l'action de l'utilisateur sur la clé ;
- une attaque triviale par rejeu est à considérer dans le cas où un attaquant hors de portée du véhicule a accès à la clé pendant quelques secondes. Pour réaliser cette attaque, un attaquant en mesure d'effectuer un ou plusieurs appuis d'ouverture/fermeture et d'enregistrer les trames émises, peut rejouer les trames capturées avant que l'utilisateur n'interagisse avec le véhicule pour ouvrir ou fermer de celui-ci. Cela est dû au fait que le RKE n'effectue pas d'échanges, mais repose uniquement sur l'envoi d'une seule trame par la clé vers l'ECU ;
- une des mesures de sécurité généralement intégrées aux systèmes RKE pour limiter les attaques par rejeu consiste à inclure des compteurs ou des codes tournants dans les trames échangées (une description détaillée de ce principe est proposée dans les paragraphes suivants). Cependant, il a été démontré dans [14] qu'une combinaison de brouillage

et de rejeu permet de compromettre un RKE implémentant ces contre-mesures. Dans ce scénario, l'attaquant contraint l'utilisateur à effectuer deux appuis consécutifs sur le bouton d'ouverture. Lors du premier appui, l'attaquant enregistre la trame et brouille la transmission simultanément. Lors du second appui, l'attaquant fait de même et rejoue le premier appui. Ainsi, le véhicule s'ouvre, mais l'attaquant dispose d'un enregistrement du second appui qui sera accepté par l'ECU lors d'un rejeu ultérieur.

Principe et évolutions Les systèmes RKE sont utilisés dans différents domaines d'application : portails, porte de garage ou d'entrée ou encore pour contrôler l'accès aux véhicules. L'utilisation d'un système RKE est généralement accompagnée d'un système d'ouverture mécanique qui permet à l'utilisateur d'ouvrir/fermer l'objet cible manuellement en cas de défaillance du système RKE.

Dans le monde automobile, les systèmes d'ouverture à distance procèdent généralement à une identification ou une authentification de la clé par l'ECU via des transmissions radiofréquences unidirectionnelles à courte portée. À leur apparition, les systèmes RKE n'utilisaient pas ou peu de sécurité : seul un code unique (généré en usine) par voiture ou par clé était transmis lors de l'action de l'utilisateur sur la clé. La technologie a néanmoins bien évolué depuis afin de prendre en considération les menaces et le modèle d'attaquant précédemment décrits.

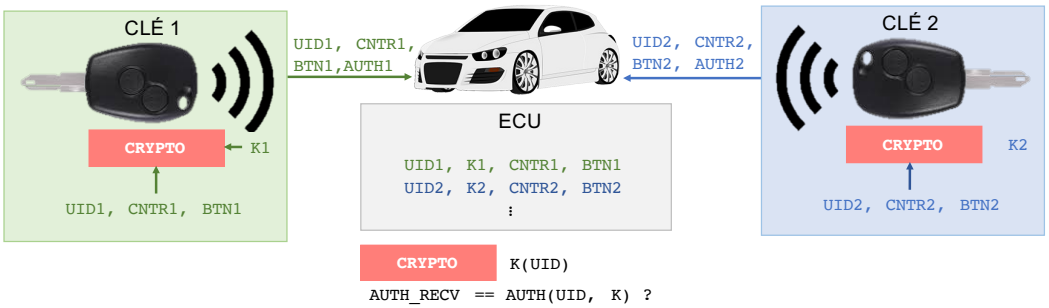

Fig. 1. Implémentation générique d'un système RKE

Le fonctionnement générique d'un système RKE récent est représenté sur la figure 1, le principe de base est fondé sur une technologie de codes tournants (*rolling codes*). Afin de garantir l'identification de la clé émettrice, un identifiant UID unique est attribué au transpondeur en sortie d'usine. Cet UID est envoyé dans la trame d'authentification lors

d'un appui sur un bouton de la télécommande. Les véhicules sont capables de gérer plusieurs clés, chaque transpondeur parmi N dispose alors de son identifiant $UID_{i,1\leq i\leq N}$. Afin de garantir un anti-rejeu des trames radio, un compteur $CNTR$ est utilisé : il est incrémenté à chaque appui de bouton, et envoyé dans la trame. Un identifiant de bouton BTN est également envoyé afin que la voiture détermine quel bouton a été appuyé sur la clé identifiée. Afin de rendre ce système sécurisé, il est nécessaire d'envoyer un authentifiant permettant de lier tous ces éléments pour que la voiture s'assure que c'est bien le transpondeur qui lui est associé qui a forgé la trame. C'est ici qu'entre en jeu l'usage de cryptographie à clé secrète : une clé de chiffrement secrète $K_{i,1\leq i\leq N}$ est présente dans la mémoire persistante du transpondeur (en général programmée dans une EEPROM sécurisée). Une fonction cryptographique f prend alors tous les éléments envoyés et calcule un authentifiant $AUTH_i = f(UID_i, CNTR, BTN, K_i)$ qui est envoyé dans la trame radio. Un attaquant n'ayant pas connaissance de la clé secrète K_i ne pourra donc pas forger de trame valide si la cryptographie est robuste. Du côté du véhicule, une base de données de tous les $UID_{i,1\leq i\leq N}$ est gardée, avec pour chaque entrée i la clé K_i associée ainsi que la valeur courante du compteur $CNTR_i$, de même que les boutons disponibles pour la télécommande en question. Lorsque l'ECU du véhicule reçoit une trame radio :

1. Il récupère l'identifiant UID_i et le compteur $CNTR\_RECV$ (tout deux envoyés dans la trame radio) ;
2. Il cherche dans sa base de données à l'entrée associée à l'UID_i, et **vérifie** que le compteur reçu $CNTR\_RECV$ est dans une fenêtre $CNTR_i \leq CNTR\_RECV \leq CNTR_i + \Delta$; La valeur de Δ doit être raisonnablement choisie. Trop faible, elle ne permettra pas de se prémunir des appuis involontaires hors champ. Trop élevée, elle tolérerait une désynchronisation trop forte entre le transpondeur et l'ECU du véhicule. En général, une valeur de quelques dizaines à quelques centaines est utilisée. Si le compteur reçu n'est pas dans la fenêtre escomptée, une **resynchronisation** est imposée par l'ECU ;
3. Lorsque cette première vérification réussit, l'ECU calcule la valeur $AUTH_i = f(UID_i, CNTR\_RECV, BTN, K_i)$ et vérifie l'égalité de $AUTH\_RECV$ avec $AUTH_i$: si l'égalité échoue, le véhicule rejette la trame et programme une **resynchronisation**. Si l'égalité est vérifiée, le compteur $CNTR_i$ stocké dans la base de données de l'ECU est mis à jour avec $CNTR\_RECV$.

La **resynchronisation** entre la clé et la voiture a généralement lieu **en champ proche** à 125 kHz après une **ouverture mécanique**. Celle-

ci assure *a priori* que la bonne clé est utilisée. Lorsque le contact est mis, l'ECU renégocie avec la clé un compteur commun permettant une resynchronisation du système.

Bien que les grands principes présentés ici soient les mêmes d'un fabricant de transpondeurs à un autre, plusieurs éléments peuvent varier :

- les caractéristiques radio (modulation, format des trames et des paquets, codage canal, etc.) ;
- la cryptographie utilisée : jusqu'à très récemment où l'AES devient l'algorithme *de facto*, divers algorithmes **propriétaires** ont été utilisés dans les transpondeurs ;
- le fabricant du transpondeur peut laisser une certaine latitude au constructeur automobile (ou son équipementier) pour personnaliser la cryptographie, la couche radio ou la couche protocolaire.

2.3 `Hitag-2`

`Hitag-2` est un algorithme de chiffrement par flot conçu par Mikron, une société autrichienne acquise par Philips Semiconductors en 1995. Cet algorithme date de la fin des années 1990/début des années 2000 et a fait l'objet d'une phase de *reverse engineering* matériel par Wiener en 2007. Les résultats de cette analyse sont publics [31], les spécifications et du code `C` sont disponibles.

Les informations présentées dans cette section proviennent de documents et de *datasheets* publics [23, 31], ainsi que de publications académiques sur le sujet [7, 8, 13, 15–17, 20, 26–28, 31].

Notations Les notations suivantes seront utilisées par la suite.

- Le corps fini à deux éléments 0 et 1 sera noté $\mathbb{F}_2$. L'opération d'addition `XOR` sur ce corps sera notée $\oplus$. L'opération de multiplication (`ET` logique) sera notée $\&$. $\mathbb{F}_2^n$ est l'ensemble résultant du produit cartésien n fois de $\mathbb{F}_2$: $\mathbb{F}_2^n = \underbrace{\mathbb{F}_2 \times \cdots \times \mathbb{F}_2}_{n \text{ fois}}$.
- Dans la suite de l'article des chaînes de n bits, éléments de $\mathbb{F}_2^n$, sont considérées. Une chaîne x composée de n bits aura son $i^{\text{ème}}$ bit noté x_i, avec $0 \leq i < n$. La chaîne sera alors représentée par $x_0 \dots x_{n-1}$. Par exemple, en notation hexadécimale `0x01` (équivalente à `00000001`) est une chaîne de huit bits où x_0 à x_6 sont nuls, et $x_7 = 1$. La chaîne de n bits à zéro est notée $\mathtt{0}^n$ et la chaîne de n bits à 1 est notée $\mathtt{1}^n$.

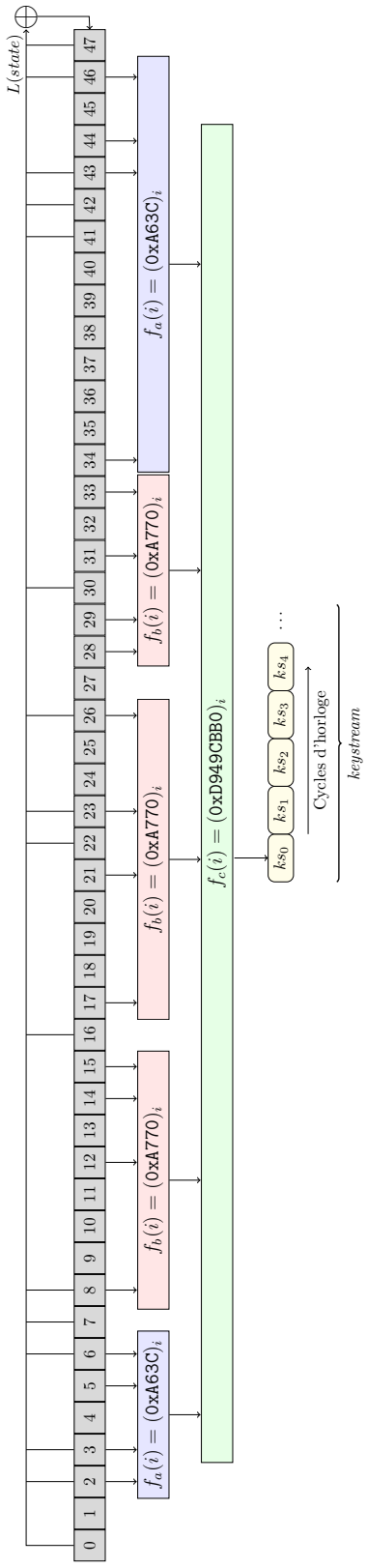

Fig. 2. `Hitag-2` : schéma synthétique de l'algorithme (source [12])

- En suivant la notation précédente, $(\texttt{0x010000})_i$ sélectionne le $i^{\text{ème}}$ bit de la chaîne de 24 bits `0x010000`. Ainsi, $(\texttt{0x010000})_7 = 1$ et $(\texttt{0x010000})_i = 0$ pour tout $0 \leq i \leq 23$ et $i \neq 7$.
- L'opération d'addition $\oplus$ entre deux chaînes de bits x et y de même taille n est définie par la chaîne de n bits résultant de l'addition bit à bit : $x \oplus y$. Ainsi $\texttt{0x0011} \oplus \texttt{0x1111} = \texttt{0x1100}$.
- L'opération de complément bit à bit $\overline{x}$ d'une chaîne x correspond à l'inversion de chaque bit de la chaîne. Ainsi $\overline{\texttt{0x01}} = \texttt{0xFE}$.
- La concaténation de deux chaînes de bits x et y, de tailles éventuellement différentes, sera notée indifféremment xy ou $x\|y$. Par exemple $\texttt{0x010}\|\texttt{0x1111} = \texttt{0x0101111}$.

Description `Hitag-2`, représenté sur la figure 2, est détaillé dans l'encadré de la figure 3. Cet algorithme est un « cousin » du `Crypto1` utilisé sur les cartes NFC Mifare Classic [9]. `Hitag-2` est basé sur une clé secrète de 48 bits, un état interne de 48 bits, une fonction linéaire agissant comme LFSR (*Linear Feedback Shift Register*) sur l'état interne, et une fonction de filtrage non linéaire f qui prend 20 bits en entrée et produit 1 bit en sortie à chaque cycle d'horloge.

Cette fonction de sortie peut être représentée comme la composition de deux niveaux différents de multiplexeurs. Les quatre bits d'entrée des fonctions f_a et f_b sont utilisés pour l'adressage et la sélection d'un des bits de données stockés dans l'état interne. Ces fonctions permettent de sélectionner 5 bits. La fonction de sortie f_c prend en entrée les données issues de f_a et f_b (5 bits) et produit 1 bit de donnée en sortie.

L'algorithme peut produire alors via son *keystream* autant de bits que nécessaire pour un chiffrement par flot. Seuls les **32 premiers bits** ks sont néanmoins utilisés comme **authentifiant** dans un contexte RKE.

Entrées de `Hitag-2` dans le cas du RKE Jusqu'à présent le fonctionnement de `Hitag-2` a été présenté comme prenant en entrées un identifiant de 32 bits id, un vecteur d'initialisation de 32 bits iv, et une clé de 48 bits k, et produisant une sortie 32 bits de *keystream* ks.

Il est nécessaire de décrire la manière dont les éléments présents dans les paquets de la trame radio s'agencent pour produire les entrées et « mapper » la sortie de l'algorithme `Hitag-2`. Cette correspondance est décrite dans l'article [12] : elle a fait l'objet de *reverse engineering* en boîte noire à **clé connue** d'un transpondeur `Hitag-2` de type `PCF7946` [22]. Cette clé était en fait partagée, dans ce cas précis, entre le RKE et le

État interne : l'état interne de 48 bits au cycle d'horloge i est noté $\alpha_i \in \mathbb{F}_2^{48}$, sa décomposition en bits est notée $\alpha_i = a_i \dots a_{47+i}$
Fonctions : une fonction linéaire et une fonction non linéaire.

- LFSR – fonction linéaire $L : \mathbb{F}_2^{48} \to \mathbb{F}_2$ telle que :
$$L(x) = x_0 \oplus x_2 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_{16} \oplus x_{22} \oplus x_{23} \oplus x_{26} \oplus x_{30} \oplus x_{41} \oplus x_{42} \oplus x_{43} \oplus x_{46} \oplus x_{47}$$
- Fonction non linéaire $f : \mathbb{F}_2^{48} \to \mathbb{F}_2$ telle que :
$$f(x) = f_c(f_a(x_2x_3x_5x_6), f_b(x_8x_{12}x_{14}x_{15}), f_b(x_{17}x_{21}x_{23}x_{26}), f_b(x_{28}x_{29}x_{31}x_{33}), f_a(x_{34}x_{43}x_{44}x_{46}))$$
Les sous fonctions non linéaires f_a, f_b et f_c sont définies par :
  - $f_a, f_b : \mathbb{F}_2^4 \to \mathbb{F}_2$ avec $f_a(i) = (\texttt{0xA63C})_i$, $f_b(i) = (\texttt{0xA770})_i$
  - $f_c : \mathbb{F}_2^5 \to \mathbb{F}_2$ avec $f_c(i) = (\texttt{0xD949CBB0})_i$

Entrées : les trois entrées de l'algorithme sont :

- un vecteur d'initialisation de 32 bits $iv \in \mathbb{F}_2^{32}$;
- une clé de 48 bits $k \in \mathbb{F}_2^{48}$;
- un identifiant de 32 bits $id \in \mathbb{F}_2^{32}$.

Fonctionnement : l'algorithme se décompose en deux phases pour faire évoluer l'état interne à chaque cycle d'horloge.

1. La phase **d'initialisation** et de **randomisation** : évolution de l'état interne initial pendant 80 cycles d'horloge en utilisant les trois entrées. Seule la fonction non linéaire f intervient.
$$\begin{array}{l} a_i = id_i,\ \forall i \in [0,31] \text{ (32 cycles d'horloge)} \\ a_{32+i} = k_i,\ \forall i \in [0,15] \text{ (16 cycles d'horloge)} \\ a_{48+i} = k_{16+i} \oplus iv_i \oplus f(a_i \dots a_{47+i}),\ \forall i \in [0,31] \text{ (32 cycles d'horloge)} \end{array}$$
2. La phase **nominale** : utilisation du LFSR sur l'état interne.
$$a_{80+i} = L(a_{32+i} \dots a_{79+i}),\ \forall i \in \mathbb{N}$$

Sortie : la sortie de l'algorithme est un *keystream* produisant un bit à chaque cycle d'horloge à partir du début de la phase nominale. Cette sortie ks est calculée à partir de l'état interne en utilisant la fonction non linéaire :

$$ks_i = f(a_{32+i} \dots a_{79+i}),\ \forall i \in \mathbb{N}$$

Le système RKE n'utilise que les 32 premiers bits de *keystream* comme authentifiant : $ks = ks_0 \dots ks_{31}$.

Fig. 3. L'algorithme Hitag-2

système d'anti-démarrage, et extraite via les attaques spécifiques décrites dans [28].

Les résultats de [12] sont confirmés dans le document [21], disponible en ligne et confirmant la décomposition des paquets émis par le transpondeur.

Nom	Longueur (bits)	Description
`SYNC`	16	Synchronisation (`0x0001`)
`UID`	32	Identifiant unique de la clé
`BTN`	4	Identifiant du bouton pressé par l'utilisateur
`CNTRL`	10	Bits de poids faibles du compteur RKE
`KS`	32	*keystream*
`CHK`	8	*checksum*

Tableau 1. Format des paquets radio du `PCF7946`

Les champs composant le paquet sont présentés dans le tableau 1. Ils correspondent aux éléments donnés dans la description générique précédente des systèmes RKE. L'identifiant unique `UID` a une longueur de 32 bits. L'identifiant de bouton `BTN` est encodé sur 4 bits (ce qui permet d'indexer 16 boutons sur une même clé). Le compteur `CNTRL` a une longueur de 10 bits, et le *keystream* `KS` de 32 bits est issu d'une fonction cryptographique f correspondant à l'algorithme `Hitag-2` décrit en détail ci-dessus. À ces éléments génériques s'ajoutent quelques spécificités :

- du côté radio : une séquence de synchronisation `SYNC` de 16 bits est envoyée en préambule afin d'assurer une bonne synchronisation d'horloge du côté du récepteur (voir détail dans la section 3 sur la radio). Un *checksum* `CHK` sur 8 bits correspondant au `XOR` de tous les octets précédents du paquet (hors octets de `SYNC`) est envoyé en postambule, ce qui permet une détection d'erreur au niveau du récepteur. Le nombre de bits des paquets de la trame n'étant pas un multiple de 8 bits (102 bits en tout), deux bits `'10'` de *padding* sont en fait ajoutés après `KS` afin que `CHK` ait un sens. Le paquet radio fait donc au final **104 bits** en comptant préambule et postambule ;
- du côté du système RKE en lui-même : le compteur de 10 bits `CNTRL` n'est en fait que la partie basse du compteur utilisé par la fonction cryptographique `Hitag-2` (le `L` signifie *low*). Une partie haute `CNTRH` (`H` pour *high*) de 18 bits est gardée **secrète** du côté de la clé émettrice et du récepteur - le calculateur ECU du véhicule. Le compteur `CNTR` réellement utilisé par la fonction cryptographique est donc sur 28 bits.

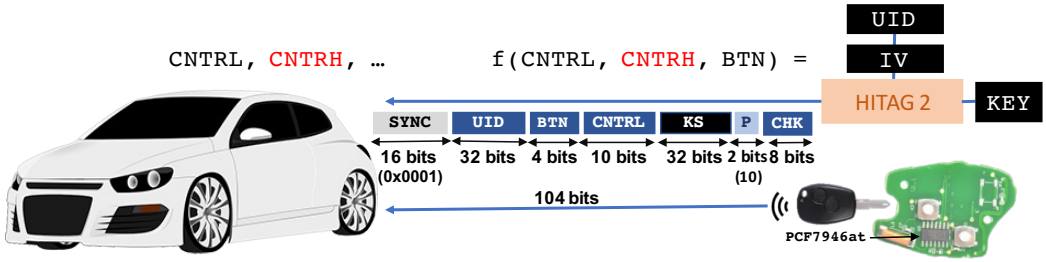

Fig. 4. Format des données envoyées par le composant PCF7946

Les éléments UID, BTN et CNTRL sont envoyés **avec le bit de poids fort en premier** [1]. Par exemple, si le compteur bas est égal à CNTRL = 1, alors la chaîne de dix bits 0000000001 est envoyée dans la trame dans cet ordre. La correspondance entre UID et id est alors immédiate. La clé k est inconnue. Il reste donc le vecteur d'initialisation iv dont les auteurs de [12] donnent la construction. Soit btn la chaîne de 4 bits extraite de BTN, soit ctr la chaîne de 28 bits forgée comme compteur en concaténant naturellement les chaînes issues des parties haute et basse CNTRH et CNTRL :

$$iv = iv_0 \dots iv_{31} = ctr \| btn = ctr_0 \dots ctr_{27} btn_0 \dots btn_3$$

Enfin, concernant la sortie de l'algorithme ks et sa correspondance avec KS, les données sont envoyées du bit 0 au bit 31 dans cet ordre, à savoir KS $= ks_0 \dots ks_{31}$. Plus de détails concernant ce point sont donnés dans la section 4.2.

Faiblesses cryptographiques de Hitag-2 et attaques L'algorithme Hitag-2 souffre de faiblesses cryptographiques identifiées depuis 2007. Tout d'abord, la clé de chiffrement est trop courte. Ensuite, la fonction de filtrage f a un faible degré d'utilisation des bits de l'état interne, ce qui se traduit par de faibles variations de ks lors de faibles variations des entrées id et iv. Enfin, comme l'id et la clé k de chaque transpondeur sont fixes, seul le vecteur d'initialisation de 32 bits iv randomise l'état interne. Or cette étape n'intervient qu'après 48 cycles d'horloge pendant lesquels seuls id et $k_0 \dots k_{15}$ sont utilisés : à l'issue de la phase d'initialisation l'état du registre interne est donc constant pour toutes les sessions d'un même transpondeur. Voici un résumé succinct des attaques liées à ces faiblesses :

[1] En fait, cet ordre exact n'est pas vraiment décrit en détail dans [12], mais nos expériences ainsi que les éléments des *datasheets* [18, 22] permettent de l'inférer.

– **Recherche exhaustive** : une clé de 48 bits est largement en dessous des standards modernes [4] (et même de ceux des années 2000). Elle rend la recherche exhaustive accessible en un temps raisonnable. Celle-ci nécessite 2 triplets (id, iv, ks) pour récupérer les 48 bits de la clé k. Notons qu'un unique triplet ne suffit pas pour retrouver de manière non ambiguë k : puisque ks fait 32 bits, il y a en moyenne $2^{48-32} = 2^{16}$ clés k produisant le même triplet (id, iv, ks) à cause des collisions possibles, le second triplet permet donc de lever cette ambiguïté. De telles recherches exhaustives ont été implémentées sur CPU, FPGA et GPU : un résumé en est donné dans le tableau 2.

Source	Plateforme	Temps
[8] (2009)	CPU 2GHz	4 ans
[26] (2011)	*Cluster* FPGA COPACOBANA (Spartan 3 - XC3S1000)	2 heures
[13] (2012)	GPU Nvidia Tesla C2050	11 heures

Tableau 2. Attaques par recherche exhaustive sur `Hitag-2`

– **Attaques algébriques** : comme pour beaucoup de systèmes de chiffrements par flot, il est possible de donner une description des liens entre la sortie ks et les entrées id, iv et k sous la forme d'équations booléennes quadratiques multivariées. Dans ces équations, les bits issus de k représentent alors les inconnues du système. L'article [8] a montré que dans le cas de `Hitag-2`, ces équations peuvent être transformées sous une forme intéressante qui simplifie le travail d'un SAT *solver*. Le temps estimé de résolution est de l'ordre 6 heures en utilisant 16 paquets à iv choisi, ou 45 heures avec 4 paquets aléatoires (l'attaque est faite sur un PC avec un CPU standard). L'article [20] étend ces attaques algébriques en utilisant `CryptoSAT`, un *solver* dédié et adapté au contexte des algorithmes cryptographiques. Peu de détails sont néanmoins donnés sur le contexte précis des attaques (paquets à iv choisi ou aléatoire). Enfin, l'article [25] fournit aussi une attaque algébrique en 6 heures, mais nécessite 50 bits de *keystream* et n'est donc pas applicable en conditions réelles.
– **Attaques « marginales »** : les attaques plus marginales comme les attaques par *Time Memory Trade Offs* [28] ainsi que les *cube attacks* [27] ne sont pas détaillées dans cet article. En effet, les attaques par *Time Memory Trade Offs* nécessitent un contexte particulier où un oracle de *keystream* fournit une sortie de taille élevée et les *cube attacks*

requièrent 500 traces générées à partir d'*iv* choisi. Ces contraintes ne sont donc pas applicables dans le contexte de cet article.

– **Cryptanalyse** : la première véritable cryptanalyse ne nécessitant pas de contrôler les *iv*, d'acquérir un nombre de traces démesuré, ou d'avoir un nombre de bits de *keystream* élevé, a été introduite dans [28]. Les auteurs réussissent à attaquer le système anti-démarrage Hitag-2 de plusieurs véhicules grâce à la récupération de 136 trames en champ proche 125 kHz. Le seul élément requis est la connaissance de l'identifiant *id* du transpondeur attaqué. Cette attaque exploite les faiblesses précédemment décrites afin de filtrer des clés candidates. Elle est réalisable en pratique, mais requiert la connaissance de l'*id* du transpondeur attaqué. L'information n'étant pas transmise dans les trames du protocole d'anti-démarrage, il faut la récupérer via un autre canal (par exemple en observant des trames RKE). La réalisation de cette attaque prend 5 minutes sur un ordinateur portable.

En conclusion, les seules attaques réalistes qui semblent adaptées au contexte RKE (où seule une communication unidirectionnelle existe, et les *iv* ne sont pas maitrisés par l'attaquant) semblent être la cryptanalyse de [28][2] et la recherche exhaustive (qui reste assez coûteuse dans l'absolu). Les auteurs de [12] estiment – à raison – qu'attendre 136 appuis de la victime sur les boutons de la clé n'est pas réaliste. Ils présentent donc une nouvelle **cryptanalyse par corrélation** qui nécessite **4 à 8 trames** et de l'ordre d'une **dizaine de minutes** sur un ordinateur portable : cette cryptanalyse est présentée ci-après.

De plus, les auteurs présentent une méthode intéressante permettant de récupérer plusieurs trames en une fois. Afin de forcer l'utilisateur légitime du véhicule à appuyer plusieurs fois sur le bouton de sa clé, ils perturbent le *checksum* de la trame via du brouillage. L'ECU rejette alors la trame, et l'utilisateur voyant que sa voiture ne réagit pas réappuie sur le bouton.

2.4 Cryptanalyse par corrélation de Hitag-2

Seule la « philosophie » de la cryptanalyse par corrélation de Hitag-2 est abordée dans ce paragraphe puisque celle-ci est utile à la compréhension de la suite de l'article, et permet de comprendre les obstacles et adaptations rencontrés en pratique (voir la section 5). Le lecteur intéressé pourra se référer à [12] pour de plus amples détails.

[2] Il n'est pas clair dans [28] que la forme aléatoire des *iv* intervienne dans la cryptanalyse du protocole 125 kHz d'anti-démarrage. Si c'est le cas, le contexte RKE qui utilise des *iv* peu variables d'une trace à l'autre ne semble pas compatible avec cette attaque.

L'idée principale de cette cryptanalyse est de fortement limiter l'espace de recherche exhaustive en identifiant des **clés candidates** ayant un **bon score de corrélation**, qui est défini ci-après. Si le filtrage des candidats est bien dimensionné, l'espace de recherche de 2^{48} clés est fortement réduit et permet de restreindre le temps de recherche à une dizaine de minutes sur un ordinateur portable « standard ». Plutôt que de calculer un score sur tous les bits de clé, la cryptanalyse le fait sur des clés candidates d'une taille de 16 bits à 48 bits. L'objectif est d'éviter une explosion en temps et en mémoire en « écrémant » les candidats les moins plausibles via leur score lors du passage d'une génération de n bits ($16 \leq n < 48$) à la génération de $n+1$ bits. Dans la suite de l'explication, l'attaquant dispose de plusieurs trames d'un même transpondeur correspondant à divers appuis (éventuellement sur différents boutons), autrement dit il a plusieurs triplets (id, iv, ks), la valeur de id étant évidemment la même pour tous ces triplets.

Le score d'un candidat de n bits ($16 \leq n \leq 48$) est calculé via une **corrélation** par rapport **aux bits de sortie** ks observés de l'algorithme. Prenons l'exemple des candidats de 16 bits : l'attaquant devine les 16 bits de poids faible de la clé k, à savoir $k_0 \dots k_{15}$. Ces bits interviennent lors de la phase d'initialisation de `Hitag-2`, et on les retrouve au 32$^{\text{e}}$ cycle d'horloge sur la partie « gauche » de l'état interne, à savoir $a_{32} \dots a_{32+15} = k_0 \dots k_{15}$ (voir les notations et la description de l'algorithme introduites en 2.3, ainsi que la figure 2). Ainsi, lorsque le premier bit de *keystream* ks_0 est produit, les 16 bits $k_0 \dots k_{15}$ interviennent (du moins les bits parmi ceux-ci sélectionnés par la fonction de filtrage f, soit les bits 2, 3, 5, 6, 8, 12, 14 et 15 de l'état interne). Les autres bits qui interviennent dans le calcul de ks_0 parmi les bits $a_{32+17} \dots a_{32+47}$ sont en fait issus de bits inconnus (non devinés) de la clé k. L'idée est alors de calculer une corrélation moyenne sur tous ces bits inconnus afin de mesurer l'adéquation des bits devinés $k_0 \dots k_{15}$ avec la sortie observée ks_0, simplement en moyennant les sorties possibles de la fonction de filtrage f réalisant ledit bit observé. Cela est réalisé en faisant une moyenne sur toutes les possibilités des bits $a_{32+16} \dots a_{32+47}$ de l'état interne. Ce qui du fait de la sélection uniquement d'une sous partie des bits par la fonction non linéaire f (les bits 17, 21, 23, 26, 28, 29, 31, 33, 34, 43, 44 et 46 de l'état interne), revient en fait à faire une moyenne sur $2^{20-8} = 2^{12}$ possibilités.

Ce score sur un unique bit étant très limité, le calcul est étendu aux bits $ks_1 \dots ks_{15}$. Les 16 bits de clé devinés agissent sur ces bits de *keystream* (ensuite, au-delà du cycle d'horloge 32+16, tous les bits de clé devinés sont sortis de l'état interne). Au final, le score d'un candidat clé de 16

bits noté $\widetilde{k}^{16} = \widetilde{k}_0 \dots \widetilde{k}_{15}$ est calculé au travers des bits de *keystream* sur lesquels ces bits agissent, ainsi que sur **toutes les traces** à disposition de l'attaquant. Sans donner plus de détails, le même calcul de score est étendu aux clés candidates dont n bits sont devinés $\widetilde{k}^n = \widetilde{k}_0 \dots \widetilde{k}_{n-1}$, en faisant évoluer n vers 48 et en ne gardant que les meilleurs candidats.

Pour ce faire, l'attaquant se donne un tableau de taille maximale fixée en mémoire. Pour chaque génération de candidats de taille $n \geq 16$ bits, il calcule le score de corrélation de tous les candidats possibles ayant « survécu » jusqu'à cette génération, i.e. en dérivant les candidats de la génération $n-1$ du tableau. Lorsque le tableau est plein, il supprime les candidats ayant le score le plus faible (via un *ranking* par exemple). La contrainte de ne garder que les candidats aux scores les plus élevés dans un tableau de taille fixe oriente la recherche exhaustive et diminue drastiquement l'espace de recherche. À la dernière génération (48^{e} bit), il reste alors très peu de candidats avec un score élevé et la bonne clé est repérée par vérification exhaustive de la génération correcte des ks de toutes les trames. D'après [12], une taille de 400 000 candidats dans le tableau est suffisant pour converger en dix minutes vers la bonne clé.

Un détail important a jusqu'à présent délibérément été omis de la description de cette cryptanalyse : jusqu'à présent l'attaquant avait en sa possession un ensemble de triplets (id, iv, ks) issus de trames radio. Or la sous-section en 2.3 précise que la partie « haute » `CNTRH` du compteur n'était pas envoyée par le transpondeur. L'attaquant ne peut donc pas complètement calculer iv, et de ce fait la cryptanalyse devient complexe puisque la partie haute fait 18 bits, ce qui induit une explosion combinatoire si toutes les possibilités d'iv sont testées.

Les auteurs de [12] présentent une manière de contourner le problème : *a priori*, en sortie d'usine, `CNTRH` est mis à zéro. Sachant que `CNTRH` est incrémenté tous les 1024 appuis à chaque débordement des 10 bits de `CNTRL`, la valeur `CNTRH` est **vraisemblablement faible et peut être inférée en fonction de l'âge du véhicule**. Ainsi, il suffit à l'attaquant de relancer sa cryptanalyse avec des iv forgés selon quelques hypothèses plausibles de `CNTRH` pour trouver la bonne clé. Le temps de cryptanalyse restera de l'ordre de quelques dizaines de minutes si le nombre d'hypothèses est faible, ce qui est le cas sur les véhicules testés par les auteurs.

2.5 Contribution de l'article

Cet article a pour objectif de vérifier la difficulté nécessaire à appliquer expérimentalement les résultats présentés dans [12]. Cette approche a l'avantage d'identifier, à partir de l'article d'origine, les éléments dont

un attaquant a besoin pour réaliser son attaque. Les outils nécessaires ainsi que les potentielles divergences expérimentales sont ainsi évalués. Une mise en perspective de la menace réelle et une réflexion autour de contremesures réalistes sont également discutées.

Cette étude a été réalisée dans un contexte d'attaquant **boîte noire** avec un accès aux seules documentations et autres articles publics. Les attaques destructrices ou invasives comme le *reverse engineering* matériel sur le transpondeur `PCF7946` ont également été exclues. Les interactions avec la clé et le véhicule associé se sont limitées aux trames radio ou RFID.

La première difficulté a été l'obtention d'échantillons de clés de voiture utilisant le composant `PCF7946`. Sur une dizaine d'échantillons, une seule clé de voiture utilisant ce composant a été identifiée par analyse des signaux radio et validée par l'examen du PCB.

L'étape suivante a été d'isoler et d'analyser les trames radio afin de retrouver la décomposition des trames et des paquets. Après avoir mis en place une chaîne de décodage correcte, la **cryptanalyse par corrélation** présentée en 2.4 a été implémentée afin de retrouver la clé k contenue dans le transpondeur. La cryptanalyse a été implémentée en appliquant *verbatim* les spécifications décrites en [12], elle a ensuite été validée sur des trames « factices » forgées selon ces mêmes spécifications. À notre surprise, cette cryptanalyse **n'a pas fonctionné** sur les trames radio émises par le transpondeur `PCF7946` de test.

La sous-section 4.2 donne le détail du recalage cryptographique réalisé afin d'isoler les divergences observées par rapport aux spécifications présentées en [12]. Ces divergences expliquent en partie pourquoi la cryptanalyse par corrélation ne donne pas de résultat dans notre cas. Leur origine reste incertaine, il peut par exemple s'agir de personnalisation opérée par l'équipementier ou le constructeur sur le véhicule.

Une **nouvelle cryptanalyse** dérivée d'une recherche exhaustive permet malgré tout d'attaquer le système RKE `Hitag-2` avec ces différences : elle est présentée en section 5. Lors des diverses expérimentations, une autre divergence intéressante avec [12] a été mise en évidence : il existe une **détection** d'attaque sur le système RKE et une **contremesure** au niveau de l'ECU du véhicule. Les éléments mis en œuvre pour adapter notre cryptanalyse à cette contremesure sont également détaillés dans cette section.

Enfin, une évaluation de la menace ainsi que des contremesures envisageables sont présentées en section 6.

3 Analyse des signaux radiofréquences

Dans le but d'étudier la sécurité des transmissions RF (quel que soit le protocole), il est nécessaire de pouvoir acquérir (i.e. utilisation de matériel de réception RF) des signaux radiofréquences et de les traiter (i.e. démodulation, décodage, etc.). Les ressources matérielles et logicielles sont directement liées au protocole de radiocommunication que l'on souhaite étudier. Ainsi la fréquence de travail, l'occupation spectrale (mono-canal ou multi-canal à saut de fréquences), la bande passante, etc. sont autant de paramètres à considérer lors de la mise en place d'un outil d'analyse dudit protocole.

De fait, la première étape sera de trouver les caractéristiques du protocole RF à l'étude. Des informations peuvent être retrouvées dans divers documents comme la norme, la documentation technique du constructeur du modem radio ou la certification de celui-ci (par exemple les rapports d'évaluation et certification FCC). Si aucun de ces documents ni aucune étude préliminaire ne sont disponibles, il sera indispensable de caractériser les signaux à l'aide d'équipements de laboratoire comme un analyseur de spectre ou un récepteur large bande. Néanmoins, pour des protocoles très simples, l'utilisation de radio logicielle couplée à un outil de traitement du signal permettra de retrouver les caractéristiques d'un signal acquis.

Le système RKE ayant été largement étudié, il existe différents outils de traitement du signal permettant d'acquérir, démoduler et décoder des paquets. Lors de l'étude bibliographique, un certain nombre de paramètres caractéristiques des couches physique et logique des signaux RKE a été récupéré. Le tableau 3 récapitule les paramètres qui seront vérifiés expérimentalement.

Paramètres	Valeurs
`Fréquences de fonctionnement`	ISM 433 MHz
`Modulation`	ASK/FSK
`Codage canal`	Manchester
`Format de paquet`	voir tableau 1

Tableau 3. Principales informations concernant la couche physique du système étudié

Dans cette section, une méthodologie *white−box* permettant de vérifier ces caractéristiques sur des paquets RF transmis par les équipements testés est proposée après un bref rappel des notions de transmission radio et de traitement du signal.

3.1 Chaîne de transmission - rappels

La chaîne de transmission de l'information, représentée en figure 5, est généralement constituée d'un émetteur, d'un canal de transmission et d'un récepteur. L'émetteur a pour fonction de transformer une donnée binaire en vue de la transmettre au récepteur au travers du canal radioélectrique par l'émission d'ondes électromagnétiques.

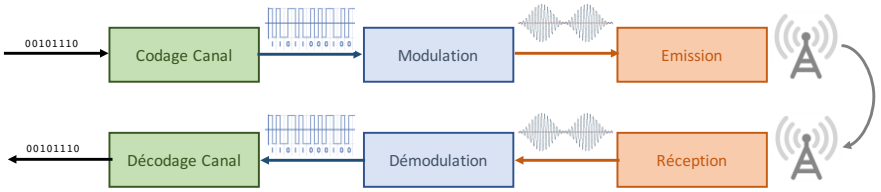

Fig. 5. Chaîne simplifiée de transmission/réception radio

Il existe deux modes de transmission. La transmission en bande de base, fondée sur la transmission directe de l'information sous forme de signal, consiste à transmettre directement le signal sur le support sans transposition de fréquence. La transmission en bande transposée, aussi appelée modulation, consiste à transmettre l'information codée en modifiant un ou plusieurs paramètres (amplitude, fréquence, phase) d'un signal sinusoïdal (appelé la porteuse). En raison des contraintes liées au canal de propagation, la plupart des transmissions radio sont réalisées en bande transposée. Afin de transmettre de l'information dans le canal radioélectrique, l'émetteur va intégrer les différentes fonctions suivantes (le récepteur intègre par réciprocité les fonctions inverses) :

- codage de canal : le codage canal vise à introduire de la redondance dans le message afin de prévenir une perte d'information due aux contraintes induites par le canal de transmission ;
- code de détection et correction d'erreur : il s'agit d'algorithmes permettant de détecter et corriger la perte de l'information due au canal de transmission ;
- modulation : transposition de la donnée binaire à la modification des caractéristiques physiques de la fréquence porteuse. En fonction du type de modulation choisie, le signal transmis sera plus ou moins robuste vis-à-vis des perturbations/dégradations induites par le canal de propagation.

3.2 Acquisition

La fréquence de fonctionnement du dispositif testé est de 433,92 MHz avec une occupation spectrale de quelques centaines de kHz (adaptation au filtre intégré au circuit d'émission). La communication est mono directionnelle. Vu les caractéristiques de la couche physique de la technologie RKE, une radio logicielle de type RTL-SDR [30] s'avère adéquate pour effectuer des acquisitions. Afin de pouvoir traiter au mieux ces signaux, la fréquence d'échantillonnage a été fixée à 2 MHz.

3.3 Démodulation

Les modulations utilisées dans le composant radio RKE sont soit la modulation d'amplitude (ASK), soit la modulation de fréquence (FSK). La configuration dépend de celle retenue par l'équipementier qui a configuré le composant RF RKE embarqué. Ainsi, il est nécessaire de vérifier la modulation retenue pour le système testé. Dans ce but, une analyse spectrale basée sur la représentation temps/fréquence de l'énergie du signal, appelée *waterfall*, est utilisée. Le résultat est proposé figure 6. Une modulation de fréquence peut être trivialement exclue car il peut être observé que le niveau d'énergie varie dans le temps sur une fréquence porteuse fixe. Il apparaît donc que la modulation utilisée pour le modèle RKE étudié est une modulation d'amplitude. Une démodulation d'amplitude est donc appliquée à ce signal.

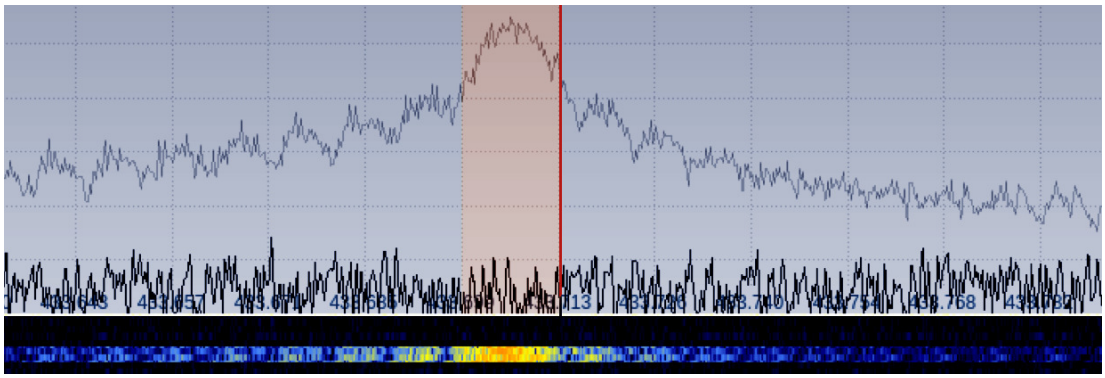

Fig. 6. Représentation type *waterfall* du signal émis par l'équipement testé

3.4 Décodage

Une fois la phase de démodulation effectuée, il est nécessaire de retrouver le type de codage de canal utilisé ainsi que le temps symbole. Le signal

obtenu après démodulation est représenté en figure 7. Une rapide analyse du signal permet de déterminer qu'une trame est composée d'un préambule de synchronisation et de deux paquets identiques successifs. Il peut être observé au début de chaque trame et de chaque paquet un bloc de synchronisation à partir duquel le temps symbole peut être mesuré (figure 8).

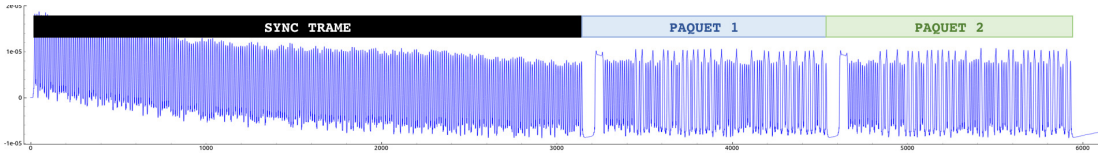

Fig. 7. Trame après démodulation

Pour l'identification du codage de canal, la connaissance de la structure des paquets (table 1) a contribué à l'élimination rapide de codages potentiels. En particulier, la composition du préambule de synchronisation des paquets et la connaissance du nombre de bits pour chaque paquet ont permis de considérer la présence très probable du codage de Manchester. Cela peut également être confirmé par la présence de transitions au maximum tous les temps symbole, ce qui réduit la probabilité d'un codage NRZ simple. En effet, vue la nature des données présentes dans les paquets (compteur, sorties d'algorithmes cryptographiques, *checksums*), il est peu probable de ne jamais avoir plus de deux bits identiques consécutifs. Une implémentation simple d'un décodeur de Manchester peut être réalisée en décodant d'abord en NRZ en divisant par deux le temps symbole (une demi-période du préambule de synchronisation), puis en appliquant un transcodage de Manchester ('`01`' code un '`0`', '`10`' code un '`1`').

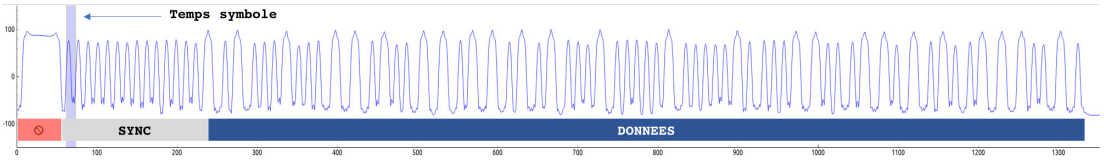

Fig. 8. Format d'un paquet de la trame. La zone surlignée correspond au temps symbole.

Le fait qu'il y ait une somme de contrôle dans chaque paquet permet de valider le décodeur implémenté.

3.5 Conclusion

Une méthode d'analyse de protocoles de communication radiofréquences a été présentée et appliquée au protocole cible. L'utilisation des informations publiquement disponibles sur le protocole cible et l'utilisation d'équipements de réception à bas coût ont permis de déterminer les caractéristiques de la couche physique du protocole. Il a notamment été déterminé que la fréquence de fonctionnement est 433,92 MHz et la bande passante de quelques centaines de kHz, ce qui a confirmé la possibilité de réaliser un récepteur avec un RTL-SDR. Il a été confirmé que le protocole repose sur une modulation d'amplitude et que le codage canal est un codage de Manchester. La composition de la trame a été déterminée et en particulier, il est intéressant de noter que dans chaque trame (à chaque appui d'un bouton de la clef) deux paquets identiques sont émis à destination du véhicule. Ces informations ont permis l'implémentation d'une chaîne de réception complète dont l'efficacité a été confirmée par le recalcul des sommes de contrôle présentes dans les paquets. Le lecteur intéressé pourra se référer à [6] qui propose une implémentation en source ouverte d'une chaîne de réception similaire basée sur GNURadio.

4 Analyse cryptographique expérimentale d'un RKE `Hitag-2`

Une fois les trames radio récupérées et décodées (en suivant la méthodologie présentée dans la section 3), l'objectif est d'extraire la clé de 48 bits k du transpondeur cible `PCF7946`. La première étape consiste à éprouver la cryptanalyse par corrélation de [12].

4.1 Implémentation de la cryptanalyse par corrélation

L'implémentation et la validation de la cryptanalyse par corrélation décrite dans la section 2.4 a d'abord été réalisée sur des paquets « factices » forgés à clés choisies selon les spécifications données en [12]. Pour simplifier les choses, le compteur « haut » `CNTRH` est fixé à zéro.

Les résultats obtenus montrent que l'implémentation réalisée fonctionne sur ces paquets, et arrive à retrouver la clé `Hitag-2` en quelques minutes tel que décrit par les auteurs. En moyenne, l'attaque par corrélation retrouve la bonne clé avec moins de 12 paquets distincts dans les deux tiers des cas. Dans certains cas dépendants de la clé k et des paquets, à savoir les valeurs des triplets (id, iv, ks), le nombre de paquets nécessaire pour que la bonne clé candidate « survive » jusqu'à la 48^{e} génération peut monter à

plusieurs dizaines. Il arrive en effet en fonction du contenu des paquets et de la clé que le bon candidat soit supprimé à une génération $n < 48$. Ceci est dû au score de corrélation sur les premiers bits de la clé devinés qui n'est pas très élevé. Ce score ne devient discriminant qu'à des générations ultérieures (mais dans ce cas il est trop tard puisque le bon candidat a déjà été éliminé).

Cette limitation de l'attaque induit une « frustration » lorsque celle-ci est appliquée sur des paquets réels dans un contexte boîte noire. Si l'attaque échoue et qu'aucune clé n'est trouvée, il est impossible de savoir si le nombre de paquets nécessaire à la cryptanalyse n'est pas suffisant ou s'il y a un problème dans l'implémentation. Les sources d'erreurs peuvent alors venir de plusieurs éléments :

- l'implémentation de `Hitag-2` : les développements réalisés ont été validés avec l'implémentation publique [31], bien que notre implémentation ait été optimisée pour que la cryptanalyse par corrélation fonctionne en temps raisonnable ;
- les entrées de l'algorithme `Hitag-2` : les spécifications de [12] ont été suivies en ce qui concerne la génération des entrées de l'algorithme à partir des paquets radio, mais il reste une zone d'interprétation, notamment sur l'ordre des bits de `UID`, `CNTRL`, `BTN` et `KS` et leurs transcriptions vers *id*, *ctr*, *btn*, *iv* et *ks* (celui-ci n'est pas précisé dans l'article) ;
- des divergences éventuelles du système observé avec celui attendu : si ces divergences sont trop importantes, le contexte d'attaque boîte noire ne permettra pas d'aller très loin dans l'analyse car le nombre d'inconnues sera trop important.

Afin de tenter de lever ces doutes, un recalage cryptographique du système RKE observé était nécessaire. La démarche suivie et les étapes pour réaliser ce recalage sont présentées en 4.2.

4.2 Recalage cryptographique et divergences

Ordre des bits des entrées et sortie de `Hitag-2` Les observations suivantes permettent de définir l'ordre des bits de `UID`, `CNTRL`, `BTN` et `KS` :

- la *datasheet* du `PCF7936` [18], proche cousin du `PCF7946`, décrit le format des paquets du protocole en champ proche permettant d'interroger le tag dans les divers modes de fonctionnement supportés par le transpondeur (par exemple pour récupérer son identifiant unique). Ce document spécifie bien que les éléments envoyés en 125 kHz le sont en

commençant par le bit de poids fort et en finissant par le bit de poids faible. Il y a de fortes chances que le même ordre soit utilisé pour le système RKE en UHF (dont les *datasheets* ne sont pas publiques). Un faisceau de présomptions nous confirme cela dans les paquets radio du PCF7946 que nous avons capturés et décodés :
  - UID a son *nibble* d'octet de poids faible qui indique la gamme de transpondeur PCF à laquelle il appartient (voir [15] pour plus de détails). Ce *nibble* a la bonne valeur si l'on considère que UID est représenté du bit 31 au bit 0 dans le paquet radio ;
  - CNTRL et BTN ont une lecture « naturelle », à savoir *big endian*, lorsqu'on les observe dans les paquets. Ainsi, nous voyons bien CNTRL s'incrémenter lors de deux appuis consécutifs avec une évolution des derniers bits concernés dans le paquet.
- concernant KS et ks, l'ordre des bits semble inversé : ks_0 est envoyé en premier et ks_{31} en dernier, ce qui est compatible avec l'ordre naturel de production du *keystream*.

Deux éléments confortent cette hypothèse pour KS :

- dans les implémentations de Hitag-2 pour Proxmark3 [1] (mode 125 KHz), le *keystream* est effectivement envoyé dans cet ordre ;
- dans nos observations, les premiers bits de KS dans les paquets radio varient peu entre des trames successives (pour lesquelles iv change de peu de bits donc). Cela tend à confirmer que ces premiers bits de KS sont les premiers bits extraits du *keystream* sur lesquels l'influence d'un petit changement d'iv est en moyenne très faible.

Malheureusement, la confirmation de ces éléments ne permet pas de converger vers une cryptanalyse par corrélation fonctionnelle sur le PCF7946 à notre disposition. La construction d'un élément important du système RKE, à savoir l'iv à partir de (CNTRL, CNTRH, BTN), n'a jusqu'ici pas été confirmée. Si celle-ci diverge de ce qui a été implémenté, la cryptanalyse ne peut clairement pas aboutir.

Pour expliquer ces divergences, il est nécessaire de trouver un moyen de faire des expérimentations à **clé connue**. Une partie de notre approche expérimentale ressemble à celle utilisée par les auteurs de [12]. Cette démarche n'est néanmoins **ni abordée ni décrite** dans leur article d'origine, mais est suggérée dans une présentation **récente** [19] dont la parution tardive n'a pas influencé les travaux que nous détaillons ci-après.

Clés vierges programmables Ces clés sont vendues de manière légale lorsqu'un propriétaire veut faire des doubles par exemple. Elles sont

programmables, et la procédure pour les rendre fonctionnelles avec un véhicule est de les « mettre à la clé » : en utilisant un mode particulier, l'ECU écrit en 125 kHz en EEPROM du transpondeur la clé cryptographique symétrique k partagée. À l'issue de cette programmation, le transpondeur est verrouillé via un ordre qui le fait transiter de manière irréversible vers le mode nominal sécurisé. Il est possible de se procurer des **clés vierges** compatibles avec notre véhicule de test (voir figure 9).

Lorsqu'un transpondeur est en mode programmable (cas des clés vierges), les informations qu'il contient sont accessibles en mode 125 kHz avec un lecteur RFID adapté (voir [15] pour une description complète). Avec le lecteur adapté, nous avons pu accéder aux pages de l'EEPROM comme l'illustre la figure 9. Il est donc possible d'obtenir `UID`, mais aussi la clé cryptographique qui est en fait une **clé usine par défaut**[3] `0x4f4e4d494b52` fournie par ailleurs dans les *datasheets* [18] et dans le code de référence `Hitag-2` [31].

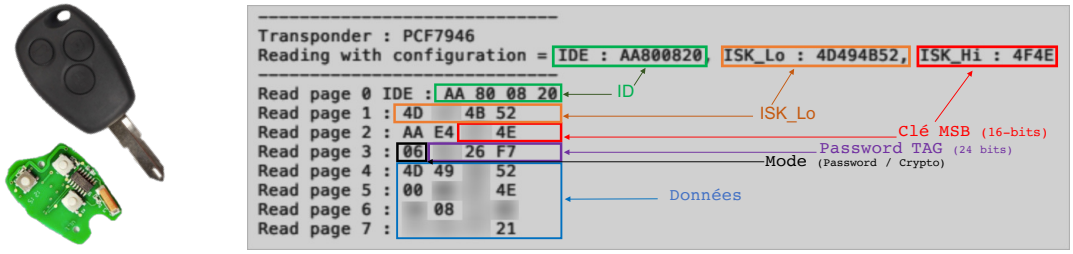

Fig. 9. Clé vierge et lecture de ses données

La connaissance de `UID` et de la clé k contenue dans le transpondeur permet de procéder au **recalage cryptographique**. Afin de limiter les inconnues en entrée, l'ordre des bits avéré de `UID` et `KS` a été utilisé comme première entrée id et sortie ks de `Hitag-2`. Puisque k est connue, les seules inconnues qui subsistent sont la valeur de iv et la valeur du compteur haut `CNTRH` (même si *a priori* en sortie d'usine celle-ci devrait être à zéro). Nous avons alors utilisé la propriété intéressante suivante :

> L'iv ne fait que 32 bits. Il est donc possible d'effectuer une **recherche exhaustive rapide** de toutes les valeurs de 32 bits permettant à id et k fixes de produire le *keystream* ks observé.

[3] Cette clé, à l'*endianness* près, peut se lire `MIKRON` en `ASCII` (nom de la société à l'origine des transpondeurs `Hitag-2` et rachetée par Philips Semiconductors en 1995).

Il y a évidemment des collisions : plusieurs *iv* peuvent produire le même *ks* aux autres paramètres fixés. Néanmoins, en ayant récupéré divers *iv* pour des paquets de plusieurs trames successives en jouant sur CNTRL et BTN, il a été possible **d'inférer une structure** de construction de *iv* à partir de ces valeurs. Cette structure est représentée sur la figure 10. Cette figure met en évidence des **divergences** par rapport à ce qui est présenté dans [12] :

- l'ordre de composition de CNTRL, CNTRH et BTN n'est pas le même ;
- un masque aléatoire MSK semble être introduit en partie basse en plus des éléments précédents. Ce masque pourrait être lié à une valeur élevée de compteur haut CNTRH, mais nous verrons en 5.3 que cela n'est pas vraiment le cas.

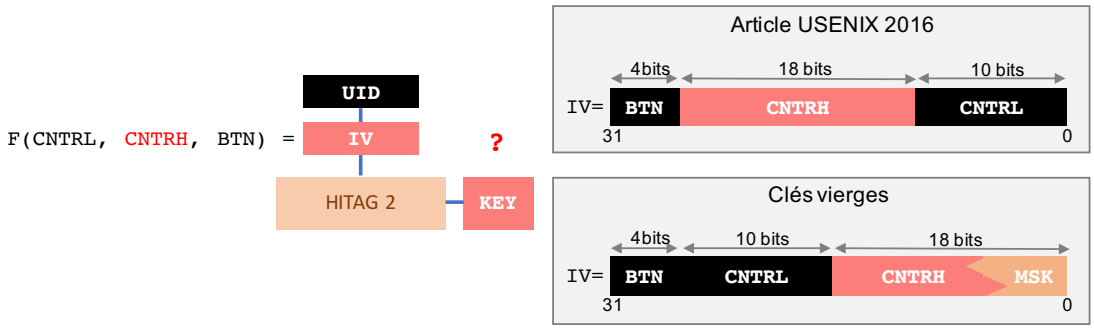

Fig. 10. Ordre des bits de l'IV utilisé en entrée de l'algorithme Hitag-2

L'ordre des bits différent entre [12] et les observations réalisées sur les clés vierges **explique pourquoi la cryptanalyse par corrélation marche mal** sur les données obtenues en pratique. En effet, dans notre cas les bits de poids faible de l'*iv* ne varient pas du tout du contenu d'une trame à une autre, alors qu'elles varient dans le cas de [12]. Or ce sont ces premiers bits de *iv* qui interviennent dans la génération des premiers bits de *ks*. Dans notre cas le fait d'avoir plusieurs trames successives n'augmente pas vraiment le score de la bonne clé candidate, qui est du coup supprimée assez tôt par l'algorithme d'« écrémage » décrit en 2.4. Par ailleurs, la présence d'un **masque non nul** en partie haute rend difficile l'utilisation de l'astuce des auteurs de [12] qui consiste à « deviner » CNTRH en fonction de l'âge de la voiture. Ce point précis ne pose en fait pas forcément de problème à cause de l'existence de clés équivalentes présentées en 5.2.

Conceptuellement, `MSK` et `CNTRH` peuvent être **fusionnés en un unique élément de 18 bits**. Le résultat de cette fusion sera **simplifié** et nommé `MSK` dans la suite de l'article.

Une explication pouvant justifier les écarts entre les deux études pourrait être que la configuration du système RKE dépend des choix techniques définis par l'équipementier, soit au niveau de bits de configuration du transpondeur, soit via une *customisation* en usine.

5 Cryptanalyse par clés équivalentes

Les propriétés identifiées lors du recalage cryptographique nous ont permis de mettre au point une nouvelle cryptanalyse. Celle-ci est fondée sur une recherche exhaustive optimisée, et tire avantage d'une spécificité de `Hitag-2`. Elle ne trouve pas la clé k enfouie dans le transpondeur, mais récupère des clés équivalentes $\widehat{k}$ qui génèrent les mêmes données que k dans une certaine fenêtre de compteur bas `CNTRL`.

Nous commençons donc en 5.1 par décrire comment nous avons optimisé une recherche exhaustive pour tenir en un temps raisonnable. Nous expliquons ensuite en 5.2 le concept de clés équivalentes, et comment elles nous permettent de forger des paquets valides. Nous décrivons ensuite en 5.3 une contremesure que nous avons découverte sur le véhicule de test, et comment notre cryptanalyse s'adapte à ce cas. Nous donnons enfin en 5.4 le *setup* matériel complet qui nous a permis d'avoir une clé factice qui envoie des paquets légitimes.

5.1 Recherche exhaustive optimisée

Une recherche exhaustive optimisée a été développée en **OpenCL** pour s'exécuter à la fois sur CPU et GPU. Cette implémentation utilise une version *bitsclicée* de `Hitag-2` qui s'y prête très bien. Elle est fortement inspirée de ce qui est présenté en [13], avec néanmoins quelques adaptations. Cet article présente en effet une implémentation OpenCL qui s'exécute en 11 heures sur un GPU Nvidia Tesla C2050 (voir le tableau 2). Notre implémentation supprime quelques optimisations spécifiques pour **gagner en scalabilité** : elle s'exécute en parallèle sur plusieurs CPU et plusieurs GPU ainsi que sur des sous-espaces de recherche de clé configurables. Nous obtenons ainsi 18 heures sur une unique Nvidia GeForce GTX780 Ti, mais descendons à **45 minutes** sur une instance Amazon EC2 avec 16 Tesla K80 et 64 CPU[4], ou **15 minutes** sur trois de ces instances en parallèle.

[4] Instance `p2.16xlarge`, voir `https://aws.amazon.com/fr/ec2/instance-types/` pour plus de détails.

À titre d'information, le coût de ces 15 minutes de calcul est d'environ 45 euros.

L'attaque par recherche exhaustive devient donc une option viable pour un attaquant qui veut faire une attaque ciblée en temps « réel » : le temps est équivalent à la cryptanalyse par corrélation. Il suffit à cet attaquant d'avoir un accès réseau et une connexion à EC2 (ou tout autre *cluster* de calcul équivalent) disponibles. Un avantage de la recherche exhaustive est qu'elle ne **nécessite que deux paquets distincts capturés** pour retrouver la clé Hitag-2.

5.2 Clés équivalentes

Nous avons discuté de l'optimisation d'une attaque par recherche exhaustive réaliste qui permet de remplacer la cryptanalyse par corrélation qui n'est pas adaptée à notre contexte.

Il subsiste néanmoins un problème *a priori* bloquant : l'attaquant ne **connaît pas** le masque de 18 bits MSK. Comme précédemment décrit, il est illusoire d'espérer effectuer 2^{18} cryptanalyses pour toutes les valeurs possibles (plusieurs centaines de milliers d'années si chaque cryptanalyse prend 10 minutes).

L'algorithme Hitag-2 possède néanmoins une propriété intéressante qui va permettre de générer des trames valides. Remarquons donc que :

> Soit $M = M_0 \dots M_{31} \in \mathbb{F}_2^{32}$ un masque de 32 bits. L'application de l'algorithme Hitag-2 avec comme entrées :
> $\Rightarrow$ identifiant id, clé k, vecteur d'initialisation iv
> **produit le même *keystream*** ks qu'avec les entrées :
> $\Rightarrow$ Identifiant id, clé $k \oplus (0^{16} \| M)$, vecteur d'initialisation $iv \oplus M$.

Il est aisé de voir pourquoi cette propriété est vraie. Si l'on se réfère à la description de Hitag-2 donnée en 2.3, durant la phase de randomisation nous avons l'équation :

$$\begin{aligned} &\forall i \in [0, 31] \text{ (32 cycles d'horloge) :} \\ &\quad a_{48+i} = k_{16+i} \oplus iv_i \oplus f(a_i \dots a_{47+i}) \\ &\qquad\quad = (k_{16+i} \oplus M_i) \oplus (iv_i \oplus M_i) \oplus f(a_i \dots a_{47+i}) \end{aligned}$$

Ainsi, les 32 bits de poids fort de la clé $k_{16} \dots k_{47}$ sont xorés avec les 32 bits du vecteur d'initialisation $iv_0 \dots iv_{31}$. Le masquage de ces deux valeurs par le même masque M sera donc compensé durant cette opération.

Considérons des données issues des paquets radio récupérés sur le transpondeur PCF7946. Ne connaissant pas les 18 bits de MSK, nous transcrivons ces données en triplets $(id, \widehat{iv}, ks)$ (que l'on nommera **équivalents**), où $\widehat{iv}$ est un vecteur d'initialisation dit équivalent au vecteur d'initialisation réel inconnu tel que $\widehat{iv} = iv \oplus (\texttt{MSK} \| 0^{14})$. Ainsi, $\widehat{iv}$ est en fait iv avec ses 18 bits de poids faible à zéro, autrement dit c'est le vecteur d'initialisation que nous pouvons immédiatement inférer des données des paquets radio capturés CNTRL et BTN sans connaître MSK.

La propriété de Hitag-2 précédemment évoquée permet d'obtenir le résultat suivant :

> Une cryptanalyse effectuée sur les triplets équivalents $(id, \widehat{iv}, ks)$ d'un transpondeur contenant une clé k permet de trouver une **clé équivalente telle que** :
>
> $$\widehat{k} = k \oplus (0^{16} \| \texttt{MSK} \| 0^{14})$$
>
> Les *keystreams* ks générés par cette clé $\widehat{k}$ sont **les mêmes** que ceux générés par k tant que MSK n'est pas modifié.

Ainsi, il suffit à l'attaquant de capturer deux paquets distincts et de lancer la cryptanalyse par recherche exhaustive optimisée présentée en 5.1 pour récupérer la clé $\widehat{k}$ en quelques minutes. Grâce à la propriété précédente, il est possible de **forger des paquets valides** grâce à la clé équivalente tant que le masque de 18 bits inconnu MSK n'est pas modifié.

MSK est en fait modifié lorsque le compteur haut est incrémenté, autrement dit tous les $2^{10} = 1024$ appuis de boutons. Nous avons donc ici une primitive permettant de **forger des paquets sur une fenêtre d'au plus 1024 appuis à partir de deux captures**. Plus exactement, si le compteur bas CNTRL vaut x, cette primitive permet de forger $(1024 - x)$ paquets qui seront considérés comme légitimes par l'ECU[5]. Il est important que les deux paquets ayant servi à la cryptanalyse **appartiennent à la même fenêtre de 1024 appuis**, car dans le cas contraire MSK est modifié ce qui a pour conséquence de faire échouer l'attaque.

Pour attaquer la fenêtre de 1024 suivant celle où la cryptanalyse a été effectuée, i.e. lorsque MSK est modifié, deux alternatives sont proposées :

1. **Inférer la nouvelle clé équivalente** : la modification de MSK correspond en fait à une **incrémentation**. Autrement dit, la nouvelle clé

[5] En fait l'attaquant peut forger 1024 paquets, les x premiers paquets étant ceux du « passé », donc inutiles à ce point pour lui, si ce n'est pour des vérifications de cohérence de *keystream* déjà généré vis-à-vis de précédentes captures.

équivalente sera en fait $\widehat{k'} \approx k \oplus (0^{16} \| (\mathtt{MSK} + 1) \| 0^{14})$ en abusant de la notation d'addition. Sachant que les effets d'une incrémentation sont le changement de n bits en cascade dans le cas de retenues chaînées, il est possible de précalculer les possibilités d'évolution de $\widehat{k}$ pour obtenir $\widehat{k'}$ (environ 18 possibilités correspondant aux cascades des $n \leq 18$ retenues). Il suffit ensuite de forger les 18 paquets possibles et de vérifier celui qui est accepté par le véhicule ;

2. **Capturer et calculer la nouvelle clé équivalente** : lors du changement de fenêtre d'appuis, l'attaquant capture un nouveau paquet $(id, \widehat{iv}, ks)$. Soit `MSK'` le nouveau masque inconnu après modification. Il suffit alors à l'attaquant d'effectuer une recherche exhaustive de 2^{18} éléments M de 18 bits, en utilisant `Hitag-2` avec la clé équivalente précédemment trouvée $\widehat{k}$, jusqu'à ce que $\widehat{iv} \oplus (M \| 0^{14})$ fournisse le *keystream* ks observé. La nouvelle clé équivalente qu'il utilisera pour cette nouvelle fenêtre de 1024 sera alors $\widehat{k'} = \widehat{k} \oplus (0^{16} \| M \| 0^{14})$. M correspond alors à la différence des masques des deux fenêtres : $M = \mathtt{MSK} \oplus \mathtt{MSK'}$.

Résumé de la cryptanalyse par clés équivalentes :

- une fois que l'attaquant a effectué sa cryptanalyse par recherche exhaustive de 2^{48} avec deux paquets distincts (15 minutes en utilisant du calcul GPU/CPU massivement parallèle, dans le *cloud* par exemple), il peut forger les paquets de la fenêtre courante ;
- au-delà, pour les fenêtres futures, il a le choix entre :
  - inférer le masque de la fenêtre suivante (incrémentation) ;
  - capturer un nouveau paquet et effectuer une recherche exhaustive de 2^{18} (quelques secondes sur un PC standard).

  Cette seconde option peut sembler moins intéressante vu qu'elle oblige à capturer un nouveau paquet radio, mais elle a l'énorme avantage de n'avoir aucune hypothèse sur l'évolution des masques entre les fenêtres de 1024 (alors que la première solution utilise les propriétés de l'incrémentation).

Ce qui est à la fois limitant et intéressant avec cette cryptanalyse, c'est que l'attaquant forge des trames valides **sans jamais utiliser la « vraie » clé k contenue dans le transpondeur**. À chaque nouvelle fenêtre de 1024, des bits de `MSK` sont en fait appris, et permettent de remonter lentement vers la vraie clé, bien que celle-ci ne soit pas nécessaire pour générer nos trames.

5.3 Contremesure ECU : resynchronisation

Lors de cette étude et des interactions avec le véhicule de test, des variations incohérentes dans l'évolution du masque de 18 bits `MSK` ont été constatées. En effet, lorsque qu'un paquet forgé est envoyé puis que le véhicule est démarré, les paquets ultérieurs générés par cryptanalyse ne sont plus reconnus comme légitimes.

L'analyse du contenu des nouvelles trames met en évidence que l'ECU **semble modifier le masque** `MSK` avec une valeur aléatoire lorsqu'une **incohérence est détectée**. Cette modification est faite lors de la resynchronisation en mode 125 kHz quand le contact est enclenché.

Une incohérence est en effet typiquement détectée lorsqu'une attaque « avec remise » est réalisée puisque :

1. L'attaquant envoie une trame forgée contenant des paquets valides qui ouvrent la voiture ;
2. L'utilisateur légitime utilise ensuite sa clé dont le compteur `CNTRL` est **inférieur au dernier compteur vu par l'ECU**.

Cette contremesure revient conceptuellement à changer de fenêtre de 1024 avec un nouveau masque aléatoire `MSK'`. Remarquons alors que notre cryptanalyse par clés équivalentes **s'adapte parfaitement à cette contremesure** lorsque l'attaquant utilise l'option de capturer une nouvelle trame pour calculer le masque de cette nouvelle fenêtre tel que présenté dans la section 5.2.

5.4 Émission de paquets valides

À l'aide des éléments présentés dans les sections précédentes, nous sommes en mesure de générer des paquets valides. Pour forger ces paquets et vérifier qu'ils sont acceptés par le véhicule de test, nous avons réalisé un outil utilisant le YARD Stick One [2].

À l'aide du *framework* RfCat [5] construit autour du circuit `CC1111` embarqué sur le YARD Stick One [11], il est possible de transmettre ou recevoir des signaux numériques sans fil à des fréquences inférieures à 1 GHz, ce qui correspond à notre cas d'usage.

Malgré le manque de documentation, le *framework* RfCat est assez simple à prendre en main. Le résultat est un clone de télécommande permettant de verrouiller ou de déverrouiller un véhicule dont le système RKE est basé sur un composant de type `PCF7946` (en tenant compte des divergences constatées par rapport à [12]).

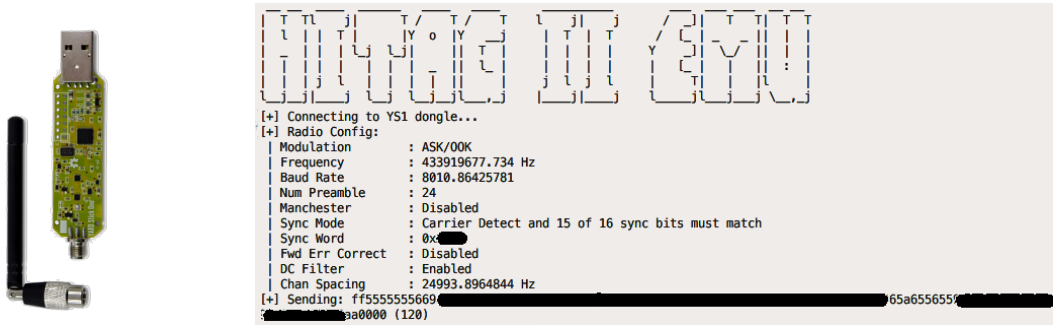

Fig. 11. YARD Stick One et simulateur de clé logiciel

6 Recommandations et contremesures

Utilisation de la cryptographie Pour limiter le risque qu'un attaquant retrouve la clé secrète ou exploite une faiblesse cryptographique, et conformément aux bonnes pratiques présentées dans [4], nous recommandons d'employer des algorithmes de chiffrement largement éprouvés dans le milieu académique.

Usage des signaux radiofréquences Des investigations dans le domaine des radiocommunications maritimes ont permis de mettre en évidence la faisabilité de détecter des émetteurs radio malveillants [3]. La caractérisation des modems RF au niveau de la couche physique permet d'observer de fortes variations des temps de montée et de descente de chaque symbole permettant de discriminer des émetteurs RF légitimes et non légitimes. Cependant, la mise en œuvre de cette technique d'identification nécessite de doter les ECU des véhicules de capacités de traitement du signal au niveau de la couche physique, ce qui peut s'avérer coûteux en calculs ou matériel dédié.

Palliatifs Le *firmware* du `PCF7946` est immuable, il n'est donc pas possible de le mettre à jour sans provoquer le rappel des véhicules vulnérables et le remplacement des composants. Dans ce contexte, nous explorons plusieurs pistes dont les objectifs sont de ralentir ou détecter de ce type d'attaque :

– la première piste vise à ralentir (ou plutôt perturber) l'attaque. Elle consiste en une amélioration de la contremesure déjà implémentée et observée sur le véhicule que nous avions à disposition. Plutôt que de modifier le masque aléatoire `MSK` lors de la détection d'une incohérence, il s'agirait de modifier celui-ci à **chaque mise sous contact/démarrage** de la voiture. Bien que cela n'empêche pas l'attaque au sens

strict, cela réduit fortement la fenêtre d'action de l'attaquant (il ne pourra mener à bien son attaque qu'entre deux démarrages). Le désavantage de cette contremesure est qu'elle oblige une écriture fréquente en EEPROM du transpondeur, ce qui peut en impacter fortement la durée de vie ;
- la seconde piste vise à détecter l'attaque. Elle consiste à observer les messages transitant sur le bus *Controller Area Network* (CAN) et à identifier les messages générés lors de l'ouverture ou de la fermeture du véhicule, de les horodater, afin d'informer l'utilisateur de la date et l'heure de la dernière ouverture. À la date de rédaction de cet article, ces travaux sont encore en cours de réalisation ;
- en complément, il est envisageable de fournir à l'utilisateur des moyens de traçabilité des événements critiques du véhicule, et en particulier dans le cas présent les journaux d'ouverture et de fermeture du véhicule. L'ajout, à destination des utilisateurs (et non plus seulement des professionnels), de systèmes de journalisation et d'interfaces de consultation, d'extraction et de gestion des journaux semble de plus en plus indispensable avec l'intégration croissante de technologies de communication et de traitement de l'information au sein des véhicules.

7 Conclusion

Dans cette étude, une analyse de résultats académiques portant sur la sécurité des transpondeurs `Hitag-2` utilisés dans un contexte RKE a été proposée. Il a été démontré la nécessité de détecter et de s'adapter à des divergences observées sur le véhicule de test utilisé lors de la réplication des expérimentations. La méthodologie décrite dans cet article a permis de concevoir et de mettre en œuvre une nouvelle cryptanalyse fondée sur une recherche exhaustive optimisée tirant avantage d'une spécificité de cet algorithme.

Il a été démontré qu'il est possible, pour un attaquant disposant d'un nombre limité de trames (au moins deux), de forger des trames d'ouverture valides en générant des clés équivalentes à la clé originale enfouie dans le transpondeur (sans retrouver cette dernière). La cryptanalyse utilisée permet, de plus, de s'adapter à une contremesure découverte durant les expérimentations sur le véhicule de test.

Ces attaques sont principalement possibles du fait de l'usage persistant d'une cryptographie faible très loin des standards modernes. Quelques recommandations et contremesures ont malgré tout été succinctement discutées, certaines font l'objet de travaux à venir.

Références

1. Code Proxmark3. Web : https://github.com/Proxmark/proxmark3.
2. YARD Stick One. Web : http://greatscottgadgets.com/yardstickone/.
3. Erwan Alincourt, Cyril Ray, Pierre-Michel Ricordel, Delphine Daré-Emzivat, and Abdel Boudraa. Méthodologie d'extraction de signatures issues des signaux ais.
4. ANSSI. RGS version 2.0 – Annexe B1, 2014. Web : https://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf.
5. ATLAS. RfCat. Web : https://bitbucket.org/atlas0fd00m/rfcat.
6. Bastian Bloessl. gr-keyfob. SDR Academy. Web : https://github.com/bastibl/gr-keyfob.
7. Nicolas T Courtois and Sean O'Neil. FSE Rump Session–Hitag 2 Cipher, 2011.
8. Nicolas T Courtois, Sean O'Neil, and Jean-Jacques Quisquater. Practical algebraic attacks on the hitag2 stream cipher. In *International Conference on Information Security*, pages 167–176. Springer, 2009.
9. Gerhard de Koning Gans, Jaap-Henk Hoepman, and Flavio D. Garcia. A Practical Attack on the MIFARE Classic. *CoRR*, abs/0803.2285, 2008. Web : http://arxiv.org/abs/0803.2285.
10. Aurelien Francillon, Boris Danev, and Srdjan Capkun. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. Cryptology ePrint Archive, Report 2010/332, 2010. http://eprint.iacr.org/2010/332.
11. Great Scott Gadgets. YARD Stick One. Web : http://greatscottgadgets.com/yardstickone/.
12. Flavio D. Garcia, David Oswald, Timo Kasper, and Pierre Pavlidès. Lock it and still lose it —on the (in)security of automotive remote keyless entry systems. In *USENIX Security*, Austin, TX, 2016. USENIX Association.
13. Vincent Immler. Breaking hitag 2 revisited. In *Security, Privacy, and Applied Cryptography Engineering*, pages 126–143. Springer, 2012.
14. Samy Kamkar. Drive It Like You Hacked It. Defcon 23. Web : https://samy.pl/defcon2015/2015-defcon.pdf.
15. Timo Kasper. *Security Analysis of Pervasive Wireless Devices–Physical and Protocol Attacks in Practice*. PhD thesis, PhD thesis. 2011. Web : https://wiki.crypto.rub.de/Counter/get.php, 2011.
16. Kerstin Lemke, Ahmad-Reza Sadeghi, and Christian Stüble. *Anti-theft Protection : Electronic Immobilizers*, pages 51–67. Springer, Berlin, Heidelberg, 2006.
17. Karsten Nohl, David Evans, Starbug Starbug, and Henryk Plötz. Reverse-Engineering a Cryptographic RFID Tag. In *USENIX security*, volume 28, 2008.
18. NXP. PCF7936AS Securing Transponders (HITAG2) datasheet. Web : http://www.mouser.com/catalog/specsheets/PCF7936AS__3851__C,1.pdf.
19. Pierre Pavlidès. Overview of some automotive RKE systems. Web : http://www.rogdham.net/media/owaspgbgday2016_pavlides_presentation.pdf.
20. Henryk Plötz and Karsten Nohl. Breaking Hitag2, HAR2009, 2009, 2011.
21. Philips Semiconductors. Immobilizer and RKE System Design (Version 4). Web : http://www.handsontec.com/pdf/Confi_Doc/Imm_Design.pdf.
22. Philips Semiconductors. PCF7946AT datasheet. Web : http://www.datasheetq.com/datasheet-download/715109/0/Philips/PCF7946.
23. Philips Semiconductors. Hitag2 protocol datasheet, 1996, 2010.
24. Craig Smith. The Car Hacker's Handbook. Web : http://opengarages.org/handbook/2014_car_hackers_handbook_compressed.pdf.

25. Mate Soos, Karsten Nohl, and Claude Castelluccia. *Extending SAT Solvers to Cryptographic Problems*, pages 244–257. Springer, Berlin, Heidelberg, 2009.
26. Petr Štembera and Martin Novotny. Breaking Hitag2 with Reconfigurable Hardware. In *Digital System Design (DSD)*, pages 558–563. IEEE, 2011.
27. Siwei Sun, Lei Hu, Yonghong Xie, and Xiangyong Zeng. Cube cryptanalysis of hitag2 stream cipher. In *International Conference on Cryptology and Network Security*, pages 15–25. Springer, 2011.
28. Roel Verdult, Flavio D Garcia, and Josep Balasch. Gone in 360 seconds : Hijacking with hitag2. In *USENIX Security*, pages 237–252, 2012.
29. Roel Verdult, Flavio D Garcia, and Baris Ege. Dismantling megamos crypto : Wirelessly lockpicking a vehicle immobilizer. In *USENIX Security*, pages 703–718, 2015.
30. Mark A Wickert and McKenna R Lovejoy. Hands-on software defined radio experiments with the low-cost rtl-sdr dongle. In *Signal Processing and Signal Processing Education Workshop (SP/SPE)*, pages 65–70. IEEE, 2015.
31. I. C. Wiener. Software optimized 48-bit Philips/NXP Mifare Hitag2 PCF7936/46/47/52 stream cipher algorithm. 2006-2007. Web : `http://cryptolib.com/ciphers/hitag2`.